Architecture to Detect, Track, and Classify Objects using LiDAR Measurements in Highway Scenarios

Mathias Pelka*
*Ibeo Automotive Systems GmbH, Germany
Reference and Tools
Email: mathias.pelka@ibeo-as.com

***Abstract*—Self-driving cars require a holistic perception of their environment. To achieve this requirement, a plethora of sensor technologies exists e.g. RGB-camera, ultra-sonic and radar. Those sensor technologies have different range, as well as resolution and behave differently with varying weather conditions. Another technology is *Light Detection and Ranging (LiDAR)*, which enables precise distance measurements. In combination with RGB-cameras, ultra-sonic, and radar, LiDAR closes the gap to enable the holistic perception of the environment.**

Due to limited experience with LiDAR sensors, there is a lack of understanding how to detect, track, and classify objects (e.g. cars, guardrails) using LiDAR data. In this paper, we propose an architecture to detect, track, and classify objects based on LiDAR measurements in highway scenarios. We evaluate our architecture using preliminary sensor data obtained from a setup including six Ibeo Lux sensors and additional a roof mounted Velodyne HDL-64E.

***Index Terms*—LiDAR, object tracking, self-driving cars, autonomous cars.**

I. Introduction and Related Work

Light Detection and Ranging (LiDAR) originated shortly after the invention of the LASER. Early applications includes meteorology or surveying [1]. LiDAR allows accurate distance measurements (e.g. accuracy is below 0.1 m for the Ibeo Lux) between a LiDAR emitter and a reflector [2]. Multiple LiDAR measurements shape a point cloud, as shown in Figure 1. The figure shows a *scan* from six Ibeo Lux LiDAR sensors as well as a scan of a Velodyne HDL-64E mounted on a car (VW Passat) in a highway scenario [2], [3]. For simplicity, the figure shows the region of interest in front of the car. Some structures are recognizable. We aim to detect, track, and classify those structures.

The scan is usually centered around the *ego vehicle* which contains the recording devices. The six Lux sensors are mounted around the bumper of the car and provide 360 degrees field of vision. In contrast to radar, LiDAR provides a higher-density point cloud, however it is more susceptible to weather. Compared to a RGB-camera, LiDAR provides also the distance towards an object.

Recent advances allow LiDAR to be used in cars, indicating a mature sensor technology. Liu et al. [4] described the measurement principle and provided a survey of different LiDAR sensors in. Object tracking using LiDAR was discussed by Fürstenberg in [5].

Object tracking for automotive applications is usually divided into two categories: *online* and *offline*. Online processing is employed for example in high level driving functions including emergency brake assist or adaptive cruise control. Additionally, online processing serves as an input for self-driving cars. Such applications require real-time processing, consequently limiting the complexity of the algorithms. An Ibeo Lux LiDAR sensor has an update rate of 25 Hz, resulting in a required processing time smaller than 40 ms [2]. Furthermore, often more than one LiDAR sensor is employed, leading to large amounts of data. In such cases, search algorithms are expensive and real-time capable tracking algorithms are required.

Offline processing is usually used to verify the performance of online algorithms. This is called *reference*. Offline processing requires large amount of data and does not need real-time capabilities. In offline processing, advanced processing is possible. This, for instance, includes tracking objects backward in time, particularly useful when only sparse LiDAR point information is available.

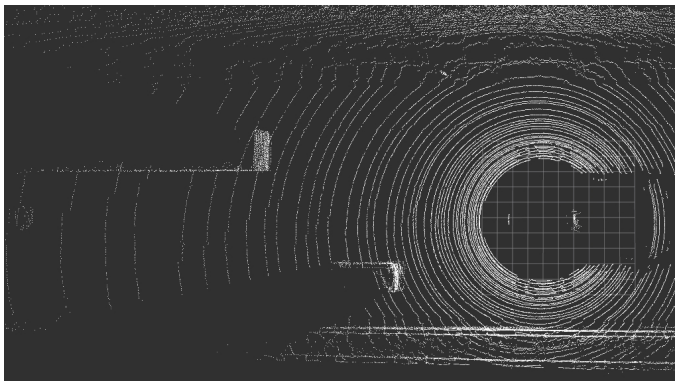

Fig. 1: Point cloud with six Ibeo Lux sensors and a roof mounted Velodyne HDL-64E. The region of interest is in front of the car.

In this paper, we discuss the relevant steps to detect, track, and classify objects based on LiDAR point cloud data in an online application for a highway scenario. The steps include ground detection, clustering, detecting best seeds (starting points for the tracking), association and Kalman-Filtering, ego motion compensation, and classification. This paper does not explain further concepts like lane detection or path planning.

The paper is structured as follows: We propose an architecture in Section II and discuss the parts of the architecture.

https://doi.org/10.24355/dbbs.084-201907151136-0

We show preliminary evaluation results in Section III using real-world data. Finally, we summarize our work in Section IV.

II. Architecture

In this section we present an architecture to detect, track, and classify objects in a highway scenario. The input for this architecture is the LiDAR point cloud, and the output are tracked objects. The architecture is shown in Figure 2 and is divided roughly in preprocessing and the tracking part.

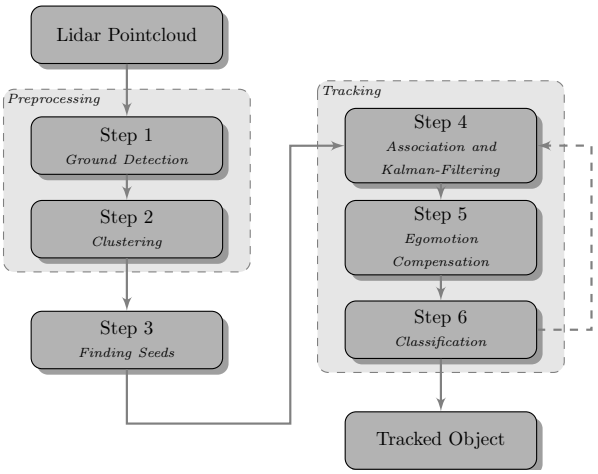

Fig. 2: Architecture to detect, track, and classify objects.

In the following subsections, we explain each block in detail.

A. Ground Detection

The ground detection is the first step in LiDAR point cloud processing and part of the preprocessing. Different techniques for ground detection are available. Ground detection labels all ground points in the point cloud as ground and consequently removes them from further processing. Ground points are included in almost every LiDAR point cloud, since they constitute the dominant plane in highway scenarios.

A simple and computationally expensive method is fitting a plane with a principle component analysis of all LiDAR points. However, this approach is not very robust, because objects, including cars or trees, will disturb the calculation. In order to remedy this, another approach is applied, where local samples of LiDAR points are taken, a plane fitted, and the plane normal vector is stored. By repeating this process and choosing different points, we determine the dominant normal vector, which is the usually the ground plane. This requires a lot of points in to be ground points. While in highway scenarios this is usually the case, in city scenarios or parking garages this may pose a problem. Here, the assumption may not be valid any longer.

In a typical scan, roughly 50 % of all points are part of the ground plane, greatly reducing the amount of data.

B. Clustering

The next step is to determine isolated objects in the point cloud. This is done using *clustering* or *segmentation* algorithms. Computation of clusters involves a distance function, which provides a measure of how far LiDAR points are apart of each other.

Standard clustering algorithms are employed for this step, including Density-Based Spatial Clustering of Applications with Noise (DBSCAN). The algorithm finds isolated clusters and requires no prior information, e.g. how many clusters are in the point cloud. Other properties of DBSCAN includes almost determinism, meaning it is independent on the processing order of LiDAR points. Furthermore, DBSCAN handles a variety of distance functions (e.g. Euclidean Distance, Manhattan Distance) to compute the distance between LiDAR points and DBSCAN has linear complexity and is therefore suited for online applications in self-driving cars [6].

The combination of ground detection and clustering is usually called preprocessing. Both are required in order to detect, track, and classify object. The result of ground detection and clustering is shown in Figure 3. Each cluster has a different color. The largest cluster is the ground which contains roughly 50 % of all LiDAR points.

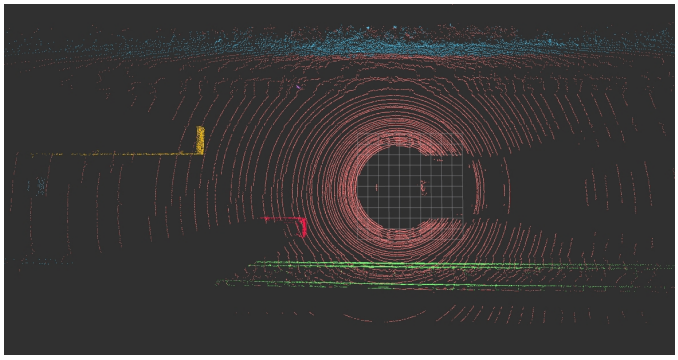

Fig. 3: Results of the preprocessing including ground detection and clustering. Each cluster has a different color. The largest cluster is the ground plane (shown in light red).

In Figure 3 three structures are visible, similar to Figure 1. A large object in front of the ego vehicle and two smaller objects, the structures have been correctly clustered.

C. Finding Seeds

The next step is to detect suitable seeds from the clusters. Those seeds are the starting point for the tracking. A cluster is not per se a good starting point for tracking and heuristics show that remote objects only consists of very few LiDAR data points. Those remote objects are hard to detect again, indicating bad seeds respectively clusters. Objects close to the ego vehicle are preferred; however, we like to detect objects not only close to the vehicle but within a certain distance. Further, clusters that are separated (i.e. clusters that do not have other clusters in their vicinity) are also preferred as starting points. The reason is, that clusters in a crowded vicinity (i.e. close to the ego vehicle) tend to occlude other objects. Consequently, we compute for each cluster a score, called *cluster quality*.

https://doi.org/10.24355/dbbs.084-201907151136-0

The higher the score, the more we are sure, that the cluster is a good seed.

If a seed is found to be suitable, we start tracking and create a *tracked object*. For each tracked object, we store the position, velocity, acceleration, and the heading.

D. Association and Kalman-Filtering

In the first tracking step of the tracked object, we aim to find clusters in the LiDAR point cloud, that match the tracked object. For that, we first predict the tracked object using a motion model (e.g. free mass with constant acceleration or a unicycle model) to the timestamp of the next LiDAR point cloud. We then associate a cluster which is in the vicinity of that tracked object and compute a measurement from it. Based on the prediction, we aim to predict the expected measurement and compare it with the actual measurement during the update step. For this, we employ an *unscented Kalman-Filter (UKF)* which is able to track non-linear movement of objects [7], [8].

In our implementation, the UKF also tracks the size of the object represented as a bounding box. Additionally, we store the covariance of the state.

E. Egomotion compensation

Since the tracked object moves relative to the ego vehicle, the motion of the ego vehicle (i.e., egomotion) has to be compensated [5]. For this, the complete ego motion between two points in time is required. The ego motion stores the change of translation and rotation. The change in position is a vector $\Delta\mathbf{r} = [\Delta x, \Delta y, \Delta z]^T$. The rotation matrix is denoted as $\mathbf{R}$. If the tracked object position is $\mathbf{r}$, the ego motion compensation is written as

$$\mathbf{r}_{\text{new}} = \mathbf{R}\left(\mathbf{r}_{\text{old}} + \Delta\mathbf{r}\right). \tag{1}$$

F. Classification

In order to optimally track an object, some additional information has proven beneficial. For instance, a car moves differently than a pedestrian. A pedestrian moves more in the sense of a free mass, rather than a car, which has certain restrictions in terms of motion. For instance, a car cannot turn on the spot, while a pedestrian can.

Different approaches for classification exist. Neural networks are popular, however, they require intensive training. Another approach based on Dempster-Shafer belief propagation was proposed by Magnier et al. [9]. A very simple approach is a decision tree, which is able to classify object, using velocity and the size of the bounding box.

After classification, the tracking process continues until no further associations have been found. In such cases, the UKF predicts the object while a confidence metric decreases. If the confidence becomes too low, the object tracking stops and the tracked object is stored.

III. Preliminary Results

We show preliminary results in this section. The data was recorded using a VW Passat with six Ibeo Lux LiDAR sensors mounted around the bumper and a Velodyne HDL-64E mounted on the roof. A visualization of the data is shown in Figure 4. It shows a similar scenario as shown in Figure 1 and Figure 3. This time, objects have been tracked (a truck and two cars) in front of the ego vehicle.

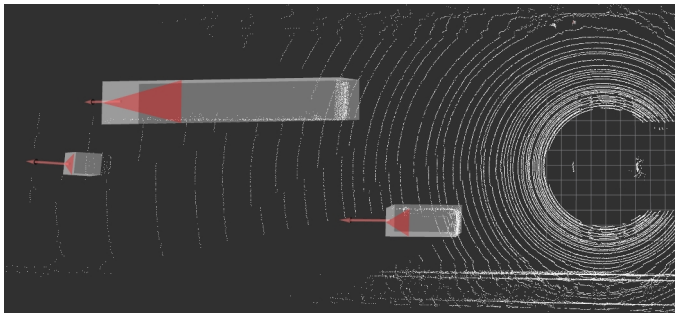

Fig. 4: Visualiazation of preliminary results

As this is still a work in progress, no *key performance indicators (KPI)* have been calculated. Examples for KPIs include accuracy, precision and the elements of a confusion matrix.

IV. Conclusion

In this paper, we proposed an architecture to detect, track, and classify objects using LiDAR measurement in a highway scenario. We have discussed the required steps to detect an object, using ground detection and clustering and discussed an approach for object tracking using Kalman filtering in combination with a decision tree to classify objects.

In future work we continue to develop this approach to enable reference object tracking. Furthermore, we aim to include key performance indicators (KPI) calculation and compare this approach against other state-of-the-art algorithms. This enables other researchers to evaluate their algorithms and to calculate performance metrics or key performance indicators.

References

[1] G. G. Goyer and R. Watson, "The laser and its application to meteorology," *Bulletin of the American Meteorological Society*, 1963.

[2] "Reference Sensor System Ibeo Automotive Systems GmbH, Hamburg," https://www.ibeo-as.com/ibeoreference/reference-sensor-system/, (Accessed on 05/21/2019).

[3] "HDL-64E," https://velodynelidar.com/hdl-64e.html, (Accessed on 05/21/2019).

[4] J. Liu, Q. Sun, Z. Fan, and Y. Jia, "TOF Lidar Development in Autonomous Vehicle," in *3rd Optoelectronics Global Conference*. IEEE, 2018.

[5] K. C. Fürstenberg, *Fahrzeugumfelderfassung und Fußgängerschutz unter Nutzung mehrzeiliger Laserscanner*. Universität Ulm Fakultät f. Ingenieurwissenschaften u. Informatik, 2009.

[6] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Conference on Knowledge Discovery and Data Mining*, 1996.

[7] E. A. Wan and R. Van Der Merwe, "The unscented Kalman filter for nonlinear estimation," in *Adaptive Systems for Signal Processing, Communications, and Control Symposium*. IEEE, 2000.

[8] M. Pelka and H. Hellbrück, "Introduction, discussion and evaluation of recursive Bayesian filters for linear and nonlinear filtering problems in indoor localization," in *International Conference on Indoor Positioning and Indoor Navigation*. IEEE, 2016.

[9] V. Magnier, D. Gruyer, and J. Godelle, "Automotive LIDAR objects detection and classification algorithm using the belief theory," in *Intelligent Vehicles Symposium*. IEEE, 2017, pp. 746–751.

https://doi.org/10.24355/dbbs.084-201907151136-0